SIMplex PARalelo (SIMPAR): FASE I

Alfonso Reinoza*, Marianela Lentini**, Alejandro Teruel*,
Lidia Da Silva¥, Alfredo Guillén¥,Tito Lozada§, Oscar Naím*

RESUMEN

El proyecto SIMPAR tiene como objetivo desarrollar una implementación paralela, en una red de transputadores, del método SIMPLEX para resolver problemas de programación lineal aplicados a la Industria Petrolera.

Este es un proyecto de investigación y desarrollo enmarcado en un esfuerzo conjunto entre MARAVEN, filial de Petróleos de Venezuela, y la Universidad Simón Bolívar.

En este trabajo presentamos una descripción de los objetivos y logros alcanzados en la primera fase del proyecto.

1. - INTRODUCCION

Los estudios realizados por la Gerencia de Tecnología de MARAVEN, a mediados del año 1988, mostraron la importancia y potencialidad del procesamiento paralelo. De hecho existían para la fecha más de 46 empresas fabricantes de equipos con paralelismo, habiendo desempeñado la industria petrolera internacional un rol protagónico en el aprovechamiento de este tipo de tecnologías. Urgía analizar en mayor detalle:

- El estado del arte de la tecnología, tanto en lo relativo a hardware como a software;
- La relación costo-beneficio de su uso;
- La viabilidad de desarrollar aplicaciones de envergadura con las herramientas disponibles,

* Departamento de Computación y Tecnología de la Información, Universidad Simón Bolívar
** Departamento de Matemáticas, Universidad Simón Bolívar
¥ I.M.G.
§ MARAVEN S.A.

- La capacidad nacional para asimilar y aplicar a corto plazo esta tecnología.

A tal fin se eligió como estrategia desarrollar un proyecto piloto que permitiese analizar estos factores. El proyecto constaría de una aplicación cuyo desarrollo exitoso mostrase la posibilidad de producir un impacto notorio que facilitase la difusión del procesamiento paralelo en la industria.

El Modelo del Area como se denomina al modelo de la planificación mensual de la producción de la refinería de Cardón, fue la aplicación seleccionada. Este presenta de manera evidente las características de un problema apropiado para el paralelismo: volumen considerable de datos, cálculos masivos y exigencias del usuario en cuanto al tiempo de respuesta.

Este modelo es un problema de programación lineal generado vía el RPMS ("Refinery Petrochemical Modeling System") de Bonner y Moore y resuelto con el MPSX (Mathematical Programing System Extended) de IBM. La operación del modelo, desde la captación de datos hasta el análisis de resultados, se hace en un ambiente de mainframe (IBM 3090) en la modalidad "batch", creando dificultades a las altas exigencias interactivas que este tipo de problema requiere.

En enero de 1989, se inició el análisis para formular un proyecto de paralelización del Modelo del Area en arquitecturas de procesadores paralelos con memorias locales (transputadores). La esencia de este proyecto consiste en desarrollar software paralelo para resolver problemas de programación lineal i.e. el Método Simplex. La poca familiaridad con la tecnología de los transputadores planteó la necesidad de desarrollar un prototipo rápido con los siguientes objetivos:

- Introducir la tecnología a la industria, en lo relativo a:
  - Hardware,
  - Metodologías de desarrollo,
  - Herramientas y métodos de productividad.
- Validar la hipótesis de paralelización del Método Simplex para resolver el Modelo del Area.

- Utilizarlo como base para la construcción de un software de programación lineal, con la calidad de un producto de mercado.

Para lograr satisfacer estos objetivos se requerían recursos humanos especializados en las siguientes áreas:

- Paralelismo
- Programación Lineal
- Análisis Numérico

El equipo inicial lo conformaron R. Karia en Paralelismo, A. Reinoza en Programación Lineal, M. Lentini en Análisis Numérico y T. Lozada como implementador. Una revisión de la literatura en el área, mostró que el proyecto SIMPAR constituía un desarrollo pionero, que requeriría de investigación para formular un modelo conceptual del Método Simplex en paralelo y resolver aspectos como:

- La paralelización de los procesos numéricos;
- La organización de los datos del problema en las memorias locales;
- La arquitectura paralela más apropiada a la aplicación ;
- Los algoritmos de comunicación entre los transputadores.

La complejidad de estos aspectos y la necesidad de tener respuestas razonablemente rápidas a las interrogantes planteadas, condujo a una fase preliminar de desarrollo de un prototipo exploratorio según la clasificación de Riddle and Williams [7]. Para ello, se seleccionó la versión más sencilla del Método Simplex (Dantzig 1949) [2] con potencial para resolver el Modelo del Area simplificándose la plataforma de desarrollo a:

- Una arquitectura de cuatro transputadores con posibilidades muy simples de comunicación;
- El lenguaje C paralelo (versión 3L) en lugar del OCCAM (lenguaje natural para transputadores).

En marzo de 1990 se dispuso de un prototipo exploratorio, que validó la hipótesis sobre el potencial de paralelización del Método Simplex. De hecho, el tiempo de ejecucion ("elapsed time") de algunos problemas de programación lineal sobre el prototipo arrojó datos comparables al tiempo de ejecución sobre MPSX en el IBM 3090.

El próximo paso consiste en desarrollar un software de calidad de producción. Dado el carácter de proyecto de investigación y la insuficiencia de recursos humanos propios, la Gerencia de Tecnología de Maraven decidió plantear el proyecto como un punto inicial de enlace en el área de informática, entre MARAVEN y la Universidad Simón Bolívar.

Esta idea cristalizó y hoy, bajo el nombre de SIMPAR, se está ejecutando un proyecto conjunto, organizado en tres fases. La primera fase consistió en una evaluacion detallada del prototipo y elaboración de un diseño basico, especialmente en cuanto a su potencial de escalabilidad a un número mayor de transputadores y la posibilidad de reutilizar porciones del diseño o el código (o ambos) del prototipo en el desarrollo del software piloto.

En este articulo presentamos la evaluación del prototipo en la sección 2, donde se consideran aspectos paralelos, numéricos y de la ingeniería de la programación. En la sección 3 se dan algunas conclusiones y recomendaciones.

2.- EVALUACION DEL PROTOTIPO

El prototipo de SIMPAR fue evaluado a la luz de sus metas y con los criterios y tecnicas de las areas del conocimiento sobre las que se apoya

- Paralelismo,
- Analisis Numerico,
- Programacion lineal,
- Ingenieria de la programación,

Brevemente comentaremos los resultados.

2.1.- PARALELISMO

Con respecto al **paralelismo**, se evaluaron los siguientes aspectos:

- Diseño conceptual de la versión paralela del Método Simplex;
- Comunicaciones;
- Métricas de Desempeño.

Un análisis detallado de estos aspectos se encuentra en el reporte interno [5]. No obstante quisieramos resaltar que la concepción paralela del método ha sido afortunada, destacando:

- La distribución por columnas de la matriz de coeficientes en las memorias de los transputadores;
- La replicación del método Simplex en cada transputador;
- La granularidad del paralelismo.

Sin embargo la simplicidad del diseño del prototipo lo ata fuertemente a la arquitectura de cuatro procesadores en una relación centralizada (master/slave). Haber adoptado esta arquitectura y la técnica de comunicación sencilla pero limitada, ha sido un factor contrario a las posibilidades incrementales del prototipo. La figura 1 presenta los conceptos aquí planteados,

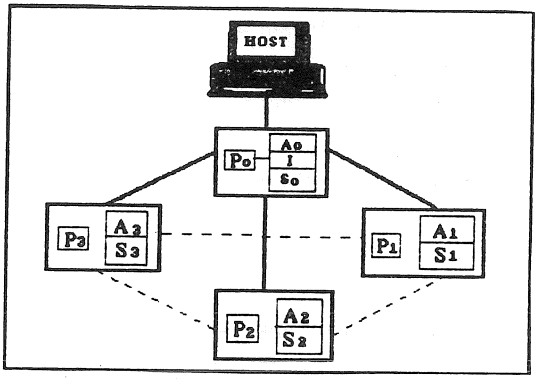

Figura 1

donde P_j denota el j-ésimo procesador, A_j el j-ésimo bloque de columnas de la matriz A, S_j la j-ésima replica del algoritmo Simplex e I representa el algoritmo coordinador de las comunicaciones.

En lo relativo a su desempeño, el prototipo fue evaluado con las siguientes métricas:

- Aceleramiento ("speedup");
- Eficiencia;

- Fracción Serial Empírica:
- Descomposicion del tiempo de ejecucion por rutina.

Dichas métricas fueron aplicadas para hallar respuestas a las siguientes preguntas:

- ¿Es aceptable el rendimiento del prototipo?
- ¿Cual sera el rendimiento del sistema con 16 transputadores?
- ¿Existen cuellos de botella o rutinas cuya mejora influyen críticamente en el rendimiento del prototipo?

2.1.1 Aceleramiento

El aceleramiento es la medida de cuánto más rápido se logra correr un algoritmo sobre N procesadores en comparación con su tiempo de ejecución en un procesador.

Idealmente uno aspira a que el aceleramiento de un algoritmo paralelo sea lineal, es decir que si se tienen N procesadores, se logrará ejecutar el algoritmo N veces más rápido que si se dispone de un solo procesador.

En la práctica el aceleramiento rara vez es lineal, por una serie de factores como:

- Tiempo adicional que se necesita para sincronizar los N procesadores o para que éstos se comuniquen entre ellos,
- El algoritmo tiene partes que deben ejecutarse serialmente o en los cuales no hay suficiente trabajo disponible para mantener ocupados los N procesadores.

En el caso del prototipo de SIMPAR, sobre 4 procesadores se obtiene un aceleramiento que se acerca a 3.5 a medida que crece el problema, en la figura 2 se muestra este comportamiento. Este es un buen resultado para un sistema de cuatro procesadores, pero ¿qué nos dice respecto al aceleramiento potencial en un sistema de dieciseis transputadores? Para hallar la respuesta a esta pregunta debe estudiarse la próxima métrica.

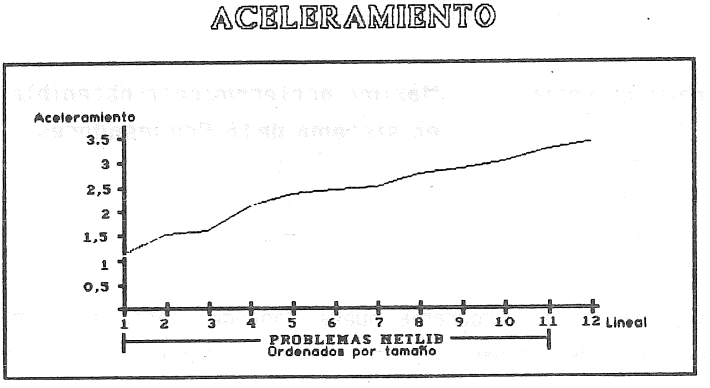

Figura 2

2.1.2 Fracción Serial Empírica

La fracción serial del programa es aquella fracción del tiempo de ejecución del programa en que la ejecucion es estrictamente serial, es decir en que el sistema se comporte como si un solo procesador estuviese ejecutando.

La fracción serial es crítica para el rendimiento de cualquier programa paralelo. En la figura 3 se ilustra la relación entre fracción serial y aceleramiento.

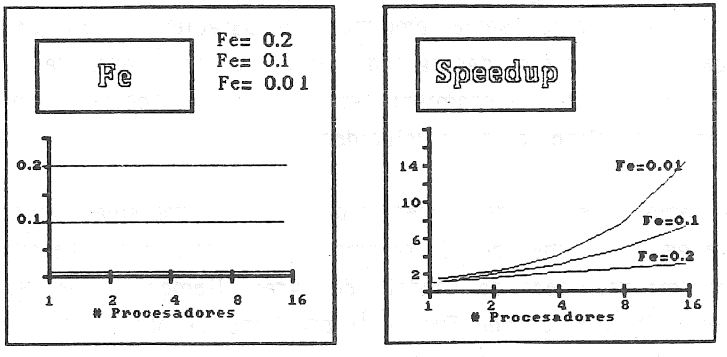

Figura 3

Se observa, como lo muestra la tabla siguiente, un incremento en el aceleramiento al disminuir la fracción serial:

Fracción serial	Máximo aceleramiento obtenible en sistema de 16 Procesadores
20%	4.0
10%	6.4
1%	13.9

La fracción serial de un programa puede depender tanto del número de procesadores como del tamaño del problema. Al aumentar el número de procesadores puede aumentar el tiempo requerido para sincronizar los procesadores; esto incrementa la fracción serial del programa. Al aumentar el tamaño del problema, la fracción serial puede aumentar (señal de posibles cuellos de botella en el algoritmo o en los recursos disponibles), puede mantenerse constante o puede incluso disminuir. El último caso (la fracción serial disminuye) es el ideal.

Dado que es sumamente difícil determinar la fracción serial teórica de un programa, se recurre a la fracción serial empírica, la cual es una aproximación experimental.

Los once problemas tomados de **Netlib** (biblioteca estándar de pruebas para cualquier paquete de Programación Lineal) muestran que el prototipo de SIMPAR tiene una fracción serial empírica que pareciera estabilizarse en 20%. Este dato por si solo es desalentador ya que de ser cierto, recorta las máximas aspiraciones de aceleramiento sobre 16 transputadores a una poco satisfactoria cuadruplicación de la velocidad.

Dado que el tamaño máximo de la matriz de los once problemas tomados de **Netlib** es apenas del orden de 10^4 elementos, se hizo una prueba adicional con un problema recortado del Modelo del Area (llamado Lineal en este reporte) cien veces más grande. Para este problema, se obtuvo una fracción serial empírica de 6%, lo cual permite proyectar un aceleramiento de 8.42 en un sistema de 16 procesadores.

¿Qué significa la discrepancia entre los dos resultados? La discrepancia es un resultado alentador ya que indica la posibilidad que la fracción serial del programa decrezca con el tamaño del problema. De resultar cierto no es descabellado pensar en la posibilidad que el Modelo del Area alcance un aceleramiento de 14 sobre un sistema de 16 transputadores.

Sin embargo conviene alertar que existen varios factores por los cuales la aseveración anterior sólo puede hacerse de manera tentativa. En particular es necesario realizar más pruebas con otros programas grandes para confirmar el resultado obtenido. Adicionalmente deben hacerse pruebas con más procesadores, ya que con los datos obtenidos es imposible usar la fracción serial empírica para estudiar posibles cuellos de botella.

En resumen, las pruebas indican que el aceleramiento obtenible sobre un sistema de 16 procesadores pueden variar desde 4 (estimado pesimista) hasta 14 (estimado optimista) con una alta probabilidad de que se obtenga un aceleramiento entre 6 y 12, para el algoritmo actual.

También se apunta hacia la necesidad de realizar pruebas adicionales sobre un prototipo sencillo de 8 y de 16 transputadores.

2.1.3 Descomposición del tiempo de ejecución por rutina

Dado que se carece de datos suficientes para aplicar la métrica anterior para detectar posibles cuellos de botella, se apeló a un estudio del consumo de tiempo de ejecución por rutina del prototipo.

Fundamentalmente hay cinco procesos ejecutándose en el prototipo de SIMPAR, un proceso maestro llamado INTERFAC que se encarga de sincronizar y coordinar a cuatro procesos esclavos llamados $SIMPLEX_i$.

El estudio de los procesos esclavos muestra una buena distribución de la carga de trabajo; casi no hay momentos en que un proceso esclavo quede ocioso esperando a que otro termine su trabajo. Adicionalmente se muestra que una de las rutinas, PIVOTEO, consume el grueso del trabajo; para ser precisos, más del 86% del tiempo de ejecución de un proceso esclavo

SIMPLEX$_1$ se lo lleva PIVOTEO, ver figura 4. Esta es una excelente noticia, ya que PIVOTEO es una rutina de cálculo que no requiere de sincronización o comunicación alguna durante su ejecución. Adicionalmente cada PIVOTEO del proceso esclavo trabaja con una partición diferente de datos, por lo que el paralelismo que se está obteniendo en PIVOTEO es muy cercano al óptimo. Ello reivindica la estrategia de paralelización, en particular la técnica de "pricing" Múltiple [6].

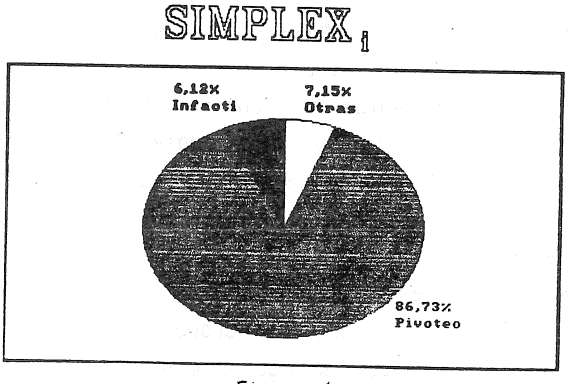

Figura 4

El estudio del proceso maestro INTERFAC muestra también una rutina, REC_SWT que consume la mayoría abrumadora del tiempo. Un análisis detallado de la misma, muestra que el tiempo mayoritario de REC_SWT corresponde a la espera por la terminación de PIVOTEO por parte de los procesos esclavos. La figura 5 ilustra esta situación.

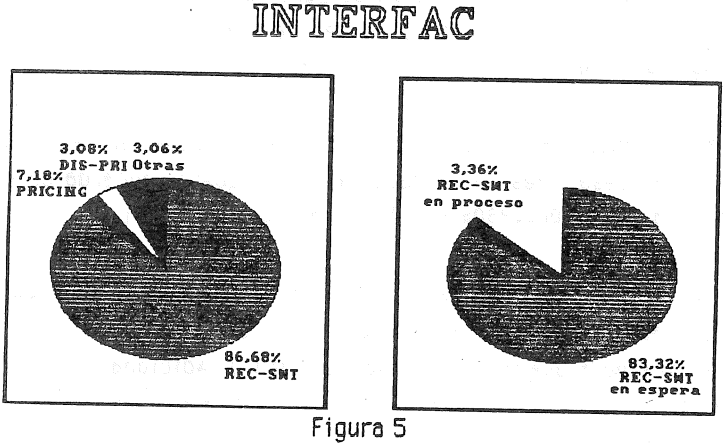

Figura 5

Todo ello indica tentativamente que los algoritmos seleccionados son los apropiados y que no existen cuellos de botella apreciables. Estos resultados por supuesto están sujetos a cambios una vez que se incorpore la funcionalidad completa del sistema de Programación Lineal, ya que el prototipo de SIMPAR no maneja adecuadamente la acumulación de errores de redondeo. Estos aspectos son importantes; a título de ejemplo, conviene recordar que muchos sistemas comerciales se ven obligados a reinvertir la base cada cincuenta iteraciones para evitar problemas de estabilidad numérica. El proceso de reinversión es costoso, y probablemente se requerirá realizarla varias veces durante la ejecución del Modelo del Area.

Adicionalmente, los resultados indican la posibilidad que las comunicaciones no lleguen a sobrepasar el 14% del tiempo de ejecución del algoritmo, incluso para topologías muy sencillas como la del anillo. Dada la simplicidad del anillo y su fácil y cómoda escalabilidad, conviene desarrollar un prototipo de 16 transputadores en anillo para estudiar el impacto de sus requerimientos de comunicación. Si el impacto es bajo, los beneficios para el sistema final son muy altos, si el impacto es alto, habrá que descartar el anillo, pero se habrán obtenido resultados valiosos para estudiar qué configuración sería la más adecuada.

2.2.- ASPECTOS NUMERICOS Y HEURISTICOS

La evaluación de los algoritmos numéricos y las heurísticas usadas en el prototipo, así como los resultados de las pruebas con los problemas de **Netlib** y algunos ejemplos aportados por la Gerencia de Planificación de MARAVEN, indican que que hubo una selección afortunada de:

- La versión original de Dantzig (tableau completo) [2] del Método Simplex;
- La técnica heurística de "Pricing" Múltiple [6];
- Los criterios heuristicos de selección del pivote.

Sin embargo, aunque estos algoritmos fueron probados, aún persisten algunos problemas:

- Estabilidad numérica. Para algunos problemas de **Netlib** el prototipo arroja resultados incorrectos y en algunos casos no los puede resolver, ver [4].

- Ciclos. Este fenómeno, que ocurre cuando el problema de programación lineal es degenerado, i.e., con frecuencia los criterios de selección del pivote no pueden resolver ambiguedades y se generan repeticiones de las mismas bases, haciendo que el programa caiga en un ciclo infinito. Para evitar esta anomalía, hemos incorporado al nuevo diseño un procedimiento heuristico anti-ciclos basado en los trabajos realizados por el grupo del Systems Optimization Laboratory de la Universidad de Stanford [3].
- Robustez: Al variar el número de procesadores o el tamaño del "pricing set", el software falla con varios problemas de **Netlib**. Esto es una clara indicación de que puede haber problemas en la implementación de las heurísticas de "pricing" y otros problemas de implementación.
- Entonación de tolerancias: Es posible que algunos de los errores que persisten en el prototipo, sean originados por los parámetros que controlan la toma de decisiones. Estas tolerancias deben ser cuidadosamente ajustadas, una tarea delicada que requiere de pruebas exhaustivas y variadas.

2.3.- INGENIERIA DE PROGRAMACION

El prototipo de SIMPAR resultó ser más un prototipo exploratorio que uno incremental. Como tal cumple sus cometidos admirablemente; demostró la factibilidad de dominar una nueva tecnología, demostró la factibilidad de paralelizar SIMPAR y dió indicios de los resultados esperables al expandir su plataforma de hardware a dieciseis transputadores. Sin embargo no fue desarrollado bajo criterios de calidad de producto final. La cohesión de algunos de sus módulos es baja, llegando en algunos casos a cohesión coincidental, el acoplamiento entre algunas rutinas es excesivamente alto.
El prototipo de SIMPAR refleja un proceso de aprendizaje y correcciones locales que conducen a un producto frágil ("brittle") respecto a su modificabilidad.

Estos resultaoos nos condujeron a rediseñar totalmente el producto de "software". El rediseño de SIMPAR siguiendo criterios de calidad, puso de manifiesto una serie de problemas no contemplados en el prototipo y mejoró

considerablemente el diseño con respecto a criterios tradicionales como se muestra en la figura 6.

CRITERIOS DE DISEÑO

		PROTOTIPO	DISEÑO ACTUAL
ACOPLAMIENTO	# Máx de Parámetros	22	5
	# de Par. de Control	7	1
	Camino máximo que recorre un dato	5	4
COHESION	# máximo de funciones entrelazadas	8	3
	Tipo de Cohesión	Coincidental (Nivel 1)	Secuencial (Nivel 6)
ENCAPSULAMIENTO		NINGUNA	Tableau, PPL (En estudio) (Todas las variables globales)
# MODULOS		20	100

Figura 6

Queda pendiente la revisión de ciertos aspectos del nuevo diseño con respecto al paralelismo (por ejemplo, como encapsular la comunicación de manera de lograr cierta independencia respecto a la configuración de hardware sin por ello perder excesivamente eficiencia).

2.4.- LAS HERRAMIENTAS

Hemos tenido la oportunidad de evaluar las herramientas de desarrollo de software para los transputadores y en general hemos encontrado que su estado es mucho más rudimentario y volátil que sus equivalentes sobre máquinas paralelas. En particular, las herramientas de apoyo a la depuración de programas son sumamente limitante, pese a los nuevos problemas que surgen en programación paralela, tales como abrazos mortales ("deadlocks").

2.5.- ANALISIS DE LA PLATAFORMA DE HARDWARE

Durante la fase 1 del proyecto se analizaron en detalle dos problemas importantes:

- Los requerimientos de memoria del modelo de área,
- La topología de interconexión entre transputadores.

Ambos problemas tienen repercusiones críticas sobre la plataforma de hardware existente. Dicha plataforma consta de:

- Dos tarjetas Quadputer, para un total de ocho transputadores con 4 Megabytes de memoria cada uno;
- Ocho transputadores con 2 Megabytes de memoria cada uno conectados en tarjetas B008 de INMOS.

Un sistema paralelo es más facil de manejar cuanto más homógeneo sea. La configuración actual rompe con la homogeneidad en tres puntos:

- Se manejan dos tecnologías de tarjetas (Microway e Inmos),
- Los transputadores tienen diferentes recursos de memoria (4Mb,2Mb),
- La red de interconexion entre transputadores es heterogenea (una conexión fija en forma de grafo completo de 4 nodos para los Quadputers y un switche reconfigurable para las B008).

Esta heterogeneidad tiene las siguientes implicaciones, de conservarse la configuración actual:

- Para manejar los requerimientos del modelo de área (unos 38 Megabytes de memoria como mínimo), es necesario implementar algún mecanismo de manejo de memoria que o bien rompa con la distribución equitativa de carga de trabajo existente actualmente o bien implemente memoria virtual (bajo esquema de cache o de disco o ambas).
- No es posible conectar los transputadores en una topología de hipercubo ni tampoco en una malla cuadrada de 4x4 transputadores.
- Debe buscarse una topología de conexión adecuada. No se plantearán aquí todas las alternativas que fueron consideradas, ellas están detalladas en la referencia [5].

Dadas las siguientes consideraciones,

- No existe un manejador de memoria virtual confiable a nivel comercial para transputadores;
- El tiempo de desarrollo de tal manejador es no trivial;
- Hacer "swapping" desde el disco del PC anfitrión degrada el sistema de manera totalmente inaceptable (estimado del orden de 24 horas adicionales de tiempo de ejecución para el Modelo del Area);
- El uso de los 4 Megabytes por transputador como "cache" degrada el sistema además de romper con la homogeneidad de distribución de carga;
- Hay indicios que las comunicaciones no son críticas y que por ende no hace falta una topología tan sofisticada como el hipercubo y puede ser que la topología de anillo baste;
- El hardware actual puede armarse en forma de anillo.

se recomienda fuertemente determinar a través del estudio de un prototipo de 16 transputadores conectados en anillo utilizando 2 Megabytes de memoria por transputador, si la topología puede limitarse a un anillo.

Dependiendo de los resultados del prototipo del anillo, existen varias opciones a tomar sobre la plataforma futura para el proyecto, que se detallan en la referencia [5].

2.6.- ASPECTOS METODOLOGICOS

Conviene insistir en que el proyecto no sólo maneja una tecnología del estado del arte (paralelismo) sino que también busca controlar la calidad del producto final así como del proceso de desarrollo mediante la aplicación de técnicas de Ingeniería de la Programación.

De esta manera, se trabaja con un modelo de desarrollo de cascada enriquecido con las siguientes nociones:

- Prototipos exploratorios e incrementales;
- Manejo del concepto de calidad;
- Actividades no limitadas al diseño de software, sino también al diseño más apropiado de la configuración de hardware;

- Inspecciones informales de tangibles;
- Manejo (sencillo) de aspectos de riesgos e incertidumbre, por ejemplo mediante la adaptación del modelo COCOMO intermedio de estimación de tiempos, ver [1].

A efecto del modelo COCOMO y a título de ejemplo, considérese el impacto de distribuir la carga heterogéneamente entre los transputadores con el objeto de cumplir con la configuración peculiar de memoria existente en la plataforma de hardware actual. Aplicando el modelo COCOMO se puede predecir que dicha distribución obligará a invertir entre 11 y 30% de trabajo adicional en el proyecto.

También es conveniente recalcar que se aplica una metodología de programación en paralelo sencilla, desarrollada en los cursos del área de Paralelismo de la Maestría en Ciencias de la Computación de la Universidad Simón Bolívar, que ha ayudado a evaluar el prototipo y sus necesidades de coordinación, identificando rápidamente potenciales cuellos de botella y concentrando los esfuerzos de análisis en esos puntos.

3.- CONCLUSIONES

De los resultados obtenidos en esta fase del proyecto, podemos concluir que el prototipo cumplió exitosamente con sus objetivos. Hoy podemos afirmar que las interrogantes planteadas en cuanto a la posibilidad de resolver el Modelo del Area de Maraven S.A., con un tiempo de respuesta aceptable, han quedado despejadas.

Los resultados en el área de paralelismo muestran con altas probabilidades un aceleramiento entre 8 y 12 sobre 16 procesadores. Además se vislumbra que una configuración de anillo sea suficiente, sin embargo, para confirmarlo se recomiendan más pruebas con el prototipo adaptado a 16 transputadores.

Se evidenció además, que la versión paralela del método Simplex implementada en el prototipo, ha sido una buena selección; no hay síntomas de cuellos de botella, la fracción serial es baja y muestra una tendencia a decrecer al aumentar el tamaño del problema.

Con respecto al análisis de los algoritmos numéricos y heurísticos, es preciso introducir procedimientos de control de errores numéricos, afinar los parámetros de control y enriquecer las heurísticas para la selección de pivotes y control de ciclos.

Sobre los aspectos metodológicos y de ingeniería de la programación, se puede concluir que se ha logrado un buen diseño (bajo acoplamiento, alta cohesión y modularidad) de la versión serial de SIMPAR. Para la versión paralela es necesario definir una notación que refleje en forma clara y precisa el paralelismo y la comunicación entre módulos.

En líneas generales, la Fase I ha mostrado la viabilidad para desarrollar un software de producción

Al margen de los aspectos técnicos, ha habido logros académicos que ameritan mención. Alrededor del proyecto se ha generado una creciente actividad de investigación que se ha manifestado en varias tesis de post-grado, entre ellas:

- Un generador de métricas de desempeño
- Apoyo a la generación de casos de prueba
- Algoritmos de programación lineal en paralelo:
  - Otras versiones del método Simplex
  - Algoritmos de puntos interiores
- Una versión paralela del algoritmo de Lemke con variables acotadas.

Conviene además mencionar que tesis de maestrías han fortalecido al proyecto SIMPAR, entre otras:

- Algoritmos de Comunicación de Hipercubos Generalizados.
- Descomposición de Dominios (Conjuntamente con INTEVEP, S.A.)

Como conclusión final, cabe destacar que los resultados de esta primera fase han sido altamente satisfactorios para ambas partes, tanto desde el punto de

vista técnico como de la visión de acercamiento entre las dos instituciones involucradas, MARAVEN y U.S.B.

4.- REFERENCIAS

1 BOEHM W., Barry: Software Engineering Economics. Prentice Hall (1981)

2 DANTZIG, George B.: Linear Programming and Extensions Princeton University Press (1963).

3 GILL, Philip E. y MURRAY, Walter y SAUNDERS, Michael A. y WRIGHT, Margaret H.: A Practical Anti-Cycling Procedure for Linear and Nonlinear Programming. Technical Report SOL 88-4 Stanford (July 1988)

4 GUILLEN, Alfredo y LENTINI, Marianela: Consideraciones Numéricas en SIMPAR (Fase I). Reporte técnico Nº IT-1991-002. Proyecto conjunto USB-MARAVEN, S.A. Departamento de Computación y Tecnología de la Información. Universidad Simón Bolívar (1991).

5 NAIM, Oscar y TERUEL, Alejandro: Consideraciones sobre el Paralelismo en SIMPAR (Fase 1). Reporte técnico interno Nº IT-1991-001. Proyecto conjunto USB-MARAVEN. Departamento de Computación y Tecnología de la Información. Universidad Simón Bolívar (1991).

6 ORCHARD-HAYS, William: Advanced Linear-Programming Computing Techniques. McGraw-Hill (1968).

7 RIDDLE, W. y WILLIAMS, L.G.: Software Environments Workshop Report. ACM SIGSOFT Software Engineering Notes, Vol. II, No. 1, p.p. 73-102, (1986).